Section III:

AMENDMENT UNDER 37 CFR §1.121 to the DRAWINGS

No amendments or changes to the Drawings are proposed.

Section IV:

AMENDMENT UNDER 37 CFR §1.121

REMARKS

Request for Telephone Interview

Applicant requests a telephone interview with the examiner following receipt of the present reply and amendment in order to answer any questions the examiner may have, and to consider any suggestions the examiner may offer. Applicant's agent, Robert H. Frantz, can be reached at 405-812-5613, by the examiner to indicate the examiner's choice of time and date for the telephone interview.

Should the examiner find the amendment and/or remarks in the present reply place the claims in a condition for allowance, this request for a telephone interview may be disregarded.

Rejections under 35 U.S.C. §103

In the Office Action, claims 1 - 30 were rejected under 35 U.S.C. §103(a) as being unpatentable over Unicode Technical Report #9 "The Bi-directional Algorithm" by David (hereinafter "BiDi #9"), in view of "Frequently Asked Questions for comp.lang.functional" by Hutton (hereinafter "Hutton FAQ" or "Hutton").

Applicant requests withdrawal of the rejections, entry of the amendment, and allowance of the claims on the following basis of law, policy or procedure:

(1) The references do not recognize, teach or suggest the problem(s) discovered by the applicant and solved by the claimed invention, and as such a *prima facie* case of obviousness has not been established. An invention may be patentable due to the discovery or identification of the problem by the inventor(s), regardless of the seeming "obviousness" of the solution to the problem disclosed by the invention, as was held by the CCPA:

> It should not be necessary for this court to point out that a patentable invention may lie in the discovery of the source of a problem even though the remedy may be

> obvious once the source of the problem is identified. This is part of the "subject matter as a whole" which should always be considered in determining the obviousness of an invention under 35 U.S.C. 103. *In re* Nomiya, 509 F.2d 566, 184 USPQ 607, 612 (C.C.P.A. 1975) (quoting *In re* Sponnoble, 405 F.2d 578, 585, 160 USPQ 237, 243 (C.C.P.A. 1969))

Further, the CCPA has held that those skilled in the art at the time of the invention would need to have knowledge of the applicant's discovery in order to know what to program a computer to do, otherwise the invention would not be obvious:

> ... Perhaps today, after reading appellants' disclosure, the public dissemination of which the patent system fosters and encourages, it might be obvious to program a general-purpose digital computer to practice the invention. But 35 U.S.C. 103 requires an analysis of the prior art at the time the invention was made to determine whether the invention was obvious ....Assuming the existence, at the time of the invention, of general-purpose digital computers as well as typical programming techniques therefor, it is nevertheless plain that appellants' invention ... was not obvious under 35 U.S.C. 103 because one not having knowledge of appellants' discovery simply would not know what to program the computer to do. *In re* Prater, 415 F.2d 1393, 162 USPQ 541.

(2) The claims include steps, elements, or limitations not taught by the references, and thus fails to support a rejection under MPEP §2143.03, which states:

> **All Claim Limitations Must Be Taught or Suggested.**
> To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art.

(3) The proposed combination would change a principle of operation of the primary reference, and thus fails to support a rejection under MPEP §2143.01, which states:

> If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious.

(4) Motivation for making the proposed combination is not provided by the cited references, and thus a *prima facie* case of obviousness has not been established, whereas MPEP §2143.01 states:

> Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art.

(5) The proposed combination depends upon a reference which does not clearly have status as prior art to the applicant's claims.

(1) The CitedReferences Do not Recognize, Teach or Solve the Problem Discovered by the Inventor

BiDi #9 is a technical report promulgated by the Unicode Consortium, which includes a textual description of a high level algorithm for positioning of characters flowing from right to left, such as Arabic or Hebrew (Pg. 1, lines 10 - 11). The high level algorithm is discussed in our background of the invention section (pg. 28, paragraph 0023). In order for a product or technology to be compliant with or compatible with this Unicode Standard, it should generally follow this algorithm.

However, as also discussed in our background of the invention section, the textual description of the algorithm is incomplete and is not comprehensive of many potential situations that may arise in processing streams of characters. We have specifically highlighted how some of the reference implementations in imperative languages, such as C and Java, fail to correctly order and display character streams under certain, specific situations and combinations of characters and codes. Actual test results for these cases have been provided in some of our tables, as well.

These problems, failures and incomplete details are not discussed or acknowledged by the BiDi #9 document, but rather was discovered by the inventor. Our test result information need not be entered into the record as evidence via an affidavit as it is part of the disclosure for which the applicant has signed a declaration, and thus has the status of evidence in the application prosecution.

(2) <u>The Proposed Combination Does Not Teach All Claimed Steps, Elements or Limitations</u>

Our invention provides a BiDi #9 compliant process and system, while also solving the problems and shortcomings of the BiDi #9 high level algorithm and the reference implementations in imperative languages which applicant has discovered.

One component of our solution is to adopt a functional programming methodology, rather than an imperative programming methodology, partly because of ease of use, modularity, etc., but in greater part in order to allow our invention to provide clear separation of the phases of assigning attributes, reordering characters, and creating the stream for display, which does not exist in the BiDi #9 reference implementations in C and Java. Our invention separates text and character reordering steps from character encoding processes (pg. 23, para. 0036), including separating the facets of layout relating to character reordering from facets related to rendering, such as line breaking, glyph selection, and shaping (pg. 25, para. 0045).

This separation of reordering and rendering steps is not taught or suggested by the BiDi #9 technical report or the Hutton FAQ. The imperative nature of the Unicode BiDi reference designs written in C and Java resulted in difficulties of handling special cases and exceptions (pg. 27, para. 0057) which we have discovered and illustrated.

To emphasize the differences in our functional language-based solution as compared to the reference designs of BiDi #9, we have amended our claims to specify separating the facets of layout relating to character reordering from facets related to rendering, such as line breaking, glyph selection, and shaping (pg. 25, para. 0045) in functional programming. Neither BiDi #9 nor Hutton teach or suggest such a solution, in imperative or functional language, nor do they recognized the problems solved by our invention.

To further emphasize the differences in our functional language-based solution as compared to the reference designs of BiDi #9, we have also amended our independent claims to specify handling the Unicode as lists of 16- or 23-bit integers as Haskell does not support Unicode directly (pg. 28, para. 0054), or "sequential runs" rather than individual characters (pg. 30, para. 0061), which is not taught by the cited references. This was originally claimed in our Claims 3, 13 and 23. In the Office Action, Claims 3, 13 and 23 were rejected using the following rationale:

> As per claim 3, the rejection of claim 1 is incorporated and further, Davis discloses that **the step of grouping characters into sequential runs using type constructors and level such that characters are processed collectively rather than individually** (p. 3:20 - 23, "a minimal set of directional formatting codes is defined to control the ordering of characters when rendered. This allows exact control of the display ordering for legible interchange and also ensures that plain text used for simple items like filenames or labels can always be (grouped into sequential runs and) correctly ordered for display"). *(pg. 5 of the Office Action, first full paragraph, bold found in original text).*

In this passage from the Office Action, it is important to note that:

(a) the bolded phrase "**the step of grouping characters into sequential runs using type constructors and level such that characters are processed collectively rather than individually**" is taken from our claim language, not from the Davis reference; and

(b) the parenthetical phrase "grouped into sequential runs and" in the quote from the Davis reference has been inserted by the Examiner and does not appear in the David reference.

Applicant traverses this representation of this passage of the Davis reference, as it appears that Davis actually uses the term "sequential runs", when Davis is actually silent as to such teaching. No evidence has been provided in the rationale for rejection to establish that the definitions of "filenames" or "labels" are equivalent to "sequential runs of integers". One of ordinary skill in the art would typically assume that filenames and labels, which are often alpha-numeric in nature, would be groups or strings of characters, not numbers. For example, Random House Webster's "Computer and Internet Dictionary", Third Edition, by Philip E. Margolis, includes these definitions:

> **filename** The name of a file. All files have names. Different operating systems impose different restrictions on filenames. Most operating systems, for example, prohibit the use of certain characters in a filename and impose a limit on the length of the filename. In addition, many systems, including DOS and UNIX, allow a filename extension that consists of one or more characters following the proper filename.
>
> **label** **1.** A name. **2.** For mass storage devices, a label is the name of a storage volume... The label provides a mnemonic name... **3.** In a spreadsheet, a label is an descriptive text placed In a cell. **4.** In programming languages, a label refers to a particular location in a program, usually a particular line of source code. ...

Absent redefinition of these terms in the references themselves, they must be interpreted in the normal, well-accepted manner. As such, BiDi #9 fails to teach handling the logically order character stream as a sequential run of integers.

(3) <u>The Proposed Combination Would Change A Principle of Operation of the Primary Reference</u>

BiDi #9 proposes a <u>line-based</u> processing approach, which inherently includes rendering steps (e.g. paragraph determination, line determination) co-mingled the steps relating to reordering:

> ... As opposed to resolution phases, this algorithm action acts **on a per-line basis**, and is **applied after any line wrapping is applied to the paragraph**. (BiDi #9 pg. 18, lines 3 - 4, emphasis added)

As discussed in the foregoing paragraphs, our invention separates the processing steps related to character ordering and layout rendering (e.g. line breaking, wrapping, paragraph breaking, etc.). Please note that our claims do not include any steps related to determining line breaks, paragraph breaks, etc.

As such, if one were to implement the BiDi #9 algorithm directly into functional language as proposed in the rationale for the rejections *without the benefit of the teachings of our disclosure*, one of ordinary skill in the art would adopt such a line-based approach, and thus would realize the inherent problems with special cases as the applicant has illustrated.

(4) Motivation to Combine BiDi #9 with Hutton FAQ is Not Found in the Cited References

The Hutton FAQ is a general reference document regarding functional language programming, and touts the general advantages of functional programming as a methodology, as one would expect from an expert in the field and author who has dedicated much of his career to promoting functional programming methodologies. Such general statements of a methodology's advantages can be found in literature for virtually any programming language, including C and Java.

In the Office Action , a statement by Hutton regarding the general advantages of functional programming has been cited as the motivation or suggestion to implement the BiDi #9 algorithm in functional language, specifically:

> "Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teaching of Hutton into the system of David to perform/implement the algorithm in a functional programming language" ... because one of ordinary skill in the and would have wanted to exploit the numerous advantages of using functional programming (e.g. algorithms written using functional programming languages are well-structured software that is easy to write, easy to debug, and provides a collection of

modules that can be re-used to reduce future programming costs)."
(Page 4 of the Office Action)

Those skilled in the art have seen similar statements of these advantages touted for C and Java, including ease of writing, structured approach, ease of debug, and modular re-use. Applicant traverses that this passage of Hutton FAQ provides actual teaching or motivation to apply functional programming language techniques to the BiDi #9 algorithm in the manner we have disclosed and claimed.

Also in the Office Action, it is proposed that Hutton generally discloses implementing algorithms in functional programming instead of imperative programming, and it has been stated that Hutton provides an example of such an implementation. However, the example is not a Unicode Bidirectional character stream rendering algorithm at all, but rather is a "task of calculating the sum of the integers from 1 to 10 ... in .. C ... using a simple loop" (pg. 3, lines 11 - 13). Hutton provides no examples of any bi-directional character processing and reordering.

(5) <u>Specific Traverse of Effective Date of Hutton FAQ.</u>

Applicant traverses the affording of an effective date of November 1, 2000, to Hutton FAQ. Applicant's agent has attempted to access the Archive.org website which appears at the bottom of the supplied printout of Hutton FAQ, and the document does not readily appear at that address. Applicant's agent has also attempted to locate the same document on the web site for the University of Nottingham without success. No other source is noted in the PTO-892 form.

The November 1, 2000, date appears to be taken from the front page of the document, which reads:

"Version of 1st November 2000"

This is not a copyright statement, nor is it an indication of when the document was actually published or made available to the public. Applicant points out that the Archive.org "frequently asked questions" page (http://www.archive.org/about/faqs.php#The_Wayback_) explains that one should submit their website in order for it to be "crawled" or archived, which may result in a delay of 6 to 12 months from the page appearing in the Archive.org archives from the date of the request. Additionally, their "technology" page (http://pages.alexa.com/company/ technology.html) describes their "ia_archiver" as "continually crawling all publicly available web sites to create a series of snapshots of the Web", which takes about 2 months to complete a complete "crawl" of the entire Internet of about 16 million web sites, and about 4.5 billion pages.

Certainly, then, the actual date of the availability of the Hutton FAQ document depended in large part on the actions of the author (e.g. when was it posted to the University of Nottingham website, and when was the ia_archiver requested to crawl it), followed by considerable delays to the actual crawling and posting of the archived copy in the Archive.org databases.

Therefore, the evidence as provided in the Office Action fails to establish that the Hutton FAQ was actually published or made available to public on November 1st, 2000, as was afforded the document in the rationale for the rejections. As the front page of the document does not purport that the document was published on 1/11/2000, it merely implies that it was last revised on this date, applicant submits that it cannot be afforded an effective date based upon this information alone.

While we maintain that Hutton FAQ fails to teach our invention regardless of its date of publication as previously discussed, if a rejection is maintained based in part or whole on the Hutton FAQ, applicant requests examiner to establish for the record the legal basis for affording this effective date to the publication.

Summary

For the reasons stated in the foregoing paragraphs, and in view of the present amendment, applicant request withdrawal of the rejections and allowance of the claims.

###